Grammars

Mattox Beckman
<beckman@iit.edu>

May 16, 2006

1 Introduction

A *grammar* is a formal specification of the syntax of a language. Its purpose is to specify the form of a legal sentence of a language. The purpose of this guide is to help the student understand these concepts as they relate to the course, so many details will be omitted. Those wanting more details should refer to one of the many good books that cover grammars, such as [ASU86].

1.1 Objectives

1. Be able to identify a correctly formed rule and grammar.
2. Be able to identify the parts of a grammar and explain their purpose. Vocabulary you need to know includes LHS, RHS, sentence, token, terminal symbol, non-terminal symbol, ϵ, rules, and parse-tree.
3. Be able to explain what makes a grammar context-free.
4. Be able to explain how a grammar and a parse-tree are related.
5. Be able to demonstrate that a grammar is ambiguous by producing two different parse-trees for the same sentence.

2 The Formal System

A grammar is an example of a formal system. To specify a formal system properly, we must describe the symbols used by the formal system, and the legal sentences.

2.1 Alphabet

There are four kinds of symbols in this system.

Non-terminal symbols will be denoted by a single capital letter, or a capital letter followed by lower-case letters. Often we will call them *non-terminals*. Examples are A, $Noun$, and $Phrase$.

Terminal symbols will be denoted by a single lower-case letter, or group of lower-case letters. Often we will call them *terminals*. Examples are x, $noun$, and id.

Arrow We have an arrow symbol, $\rightarrow$. It may be pronounced "arrow", "goes to", or "becomes".

ϵ This symbol is pronounced "epsilon".

2.2 Sentences

A sentence in this system is called a *rule*. A rule consists of a non-terminal symbol, followed by an arrow, and then either an ϵ, or a string of terminal and non-terminal symbols.

Here are some example rules:

$$\begin{aligned} S &\rightarrow x\ y\ z \\ E &\rightarrow \epsilon \\ F &\rightarrow A\ y \\ G &\rightarrow q \end{aligned}$$

The part of the rule before the arrow is called the *left-hand side* (or LHS), similarly the part of the rule after the arrow is called the *right-hand side* (or RHS). The LHS must contain exactly one non-terminal symbol. The RHS can have any combination of terminals and non-terminals, or it may have a single ϵ and nothing else.

A set of rules is called a *grammar*.

A grammar may have multiple rules with the same LHS, for example:

$$\begin{array}{lcl} S & \rightarrow & x\ E\ z \\ S & \rightarrow & \epsilon \\ E & \rightarrow & E\ y \\ E & \rightarrow & q \end{array}$$

It is customary to group rules with the same LHS together. If a LHS is repeated, it is a common shorthand notation to write the following rules omitting the LHS, and possibly replacing the arrow with the | symbol, pronounced "or". The above grammar could also be written as

$$\begin{array}{lcl} S & \rightarrow & x\ E\ z \\ & | & \epsilon \\ E & \rightarrow & E\ y \\ & | & q \end{array}$$

or as

$$\begin{array}{lcl} S & \rightarrow & x\ E\ z \\ & \rightarrow & \epsilon \\ E & \rightarrow & E\ y \\ & \rightarrow & q \end{array}$$

For a grammar to be valid, every terminal symbol in the grammar must appear on the LHS of at least one rule. The first symbol in the grammar, often called S, is known as the *start symbol*.

2.3 Recursion

A rule is said to be *recursive* if the symbol on the LHS also occurs on the RHS. The rule

$$S \rightarrow x\ S\ y$$

is recursive. If the symbol occurs in the left-most position of the RHS, then it is said to be *left-recursive*. For example, the rule $E \rightarrow E\ y$ from the grammar above is left-recursive.

3 Interpretation

The rules of a grammar G specify the legal sentences of a language. A sentence is constructed from words, or tokens, which stand by themselves; and by phrases, which themselves must be constructed from other words or phrases. The terminal symbols represent tokens, and the non-terminal symbols represent phrases.

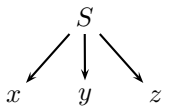
There are two ways to use grammars. A grammar can show how to construct a tree by consuming input, or it can show how to construct a tree by expanding non-terminal nodes.

One can think of a rule as a recipe. In order to construct the symbol given in the LHS, you need to supply the symbols specified in the RHS, in the proper order. Once supplied, these symbols are combined to form the symbol on the LHS.

For example, the grammar

$$S \rightarrow x\ y\ z$$

indicates that an S is generated by combining an x, a y, and a z. The result is the following tree:



The ϵ represents nothingness. An ϵ-rule indicates that the symbol is optional.

A second interpretation is that, given the non-terminal on the LHS, it can be rewritten, or expanded, to the sequence on the RHS. Again, with the example above, if you had an S, you could replace it with $x\ y\ z$.

3.1 Context-Free and Context-Sensitive

The fact that only a single non-terminal is allowed on the LHS indicates that the grammars we are considering are *context-free*. No matter where the non-terminal occurs, the rules given can be used to expand it.

In contrast, a rule such as $x\ A\ y \rightarrow q\ r$ could indicate that A may be expanded into $q\ r$, but only when surrounded by x and y. So, $x\ A\ y$ would be rewritten as $x\ q\ r\ y$. This rule would not apply to the string $x\ A\ q$, for example. Grammars that have such rules are called *context-sensitive*, and are beyond the scope of CS 440. From now on, we will use the terms *grammar* and *context-free grammar* interchangeably.

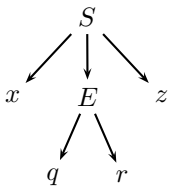
3.2 Parse Trees

Grammars are often used for parsing. The problem of parsing is to take a list of tokens as an input, and return a *parse tree* as an output.

For example, given the grammar

$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & q\ r \end{array}$$

and the input string $x\ q\ r\ y$, we would get the following parse tree.



Notice that the terminal symbols are the leaves of the tree and that the non-terminal symbols are the branches of the tree.

3.3 Validation and Ambiguity

One purpose of a grammar is to describe the set of valid sentences in a language. If a grammar can be used to create a tree out of a certain string of tokens, then the grammar is said to *accept* that string as a valid sentence in the language. Conversely, if the grammar cannot be used to generate a tree from the sentence, then we say that the grammar *rejects* that string, and that the string is not a valid sentence in the language.

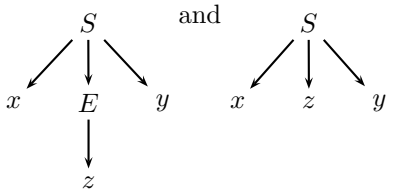
When working with grammars in the context of programming languages, we expect that most of the time a grammar will make only one tree for a valid sentence in the language. This is not required, however. It is possible to write grammars that can produce two or more different trees for the same sentence. Such grammars are called *ambiguous*. It is important to be able to recognize ambiguous grammars, because they affect the operation of the parsing algorithms we will discuss later.

Here are some examples of ambiguous grammars.

Example 1

$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ & | & x\ z\ y \\ E & \rightarrow & z \end{array}$$

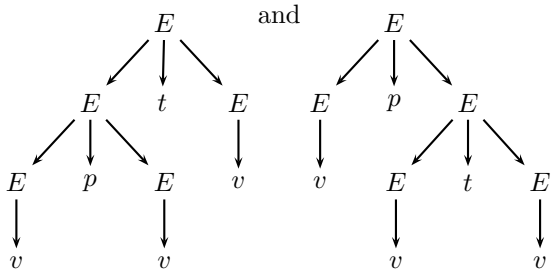
Given the sentence $x\ z\ y$, we have two parse trees:



Example 2

$$\begin{array}{lcl} E & \rightarrow & E\ p\ E \\ & | & E\ t\ E \\ & | & v \end{array}$$

Given the sentence $v\ p\ v\ t\ v$, we have two parse trees:



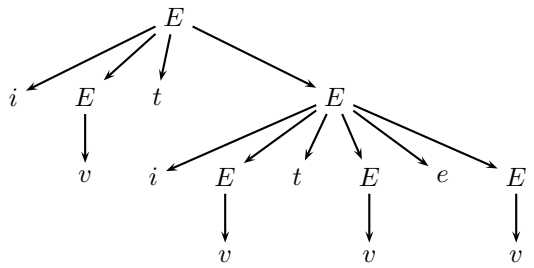
This particular example is similar to the precedence problem: given the equation $2 + 3 * 4$, do

we perform the addition first (corresponding to the first tree) or the multiplication first (corresponding to the second tree)?

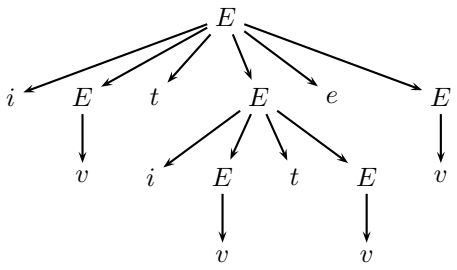
Example 3

$$\begin{array}{rcl} E & \rightarrow & i\ E\ t\ E \\ & | & i\ E\ t\ E\ e\ E \\ & | & v \end{array}$$

Given the sentence $i\ v\ t\ i\ v\ t\ v\ e\ v$, we have two parse trees:



and



This example is sometimes called the "dangling else" problem. Consider the expression `if c then if x then y else z`. The `else` clause could belong to either the first `if` or the second `if`, and the meaning of the program will be different depending on which of the two choices you pick.

4 Problems

Try these problems to see how well you understand the concepts. Solutions are at the end.

1. Which of the following are valid context-free grammar? If the grammar is invalid, explain what is wrong with it.

   (a)
   $$\begin{array}{rcl} S & \rightarrow & A\ x\ y \\ x\ A & \rightarrow & x\ y\ z \\ A & \rightarrow & q \end{array}$$

   (b)
   $$\begin{array}{rcl} S & \rightarrow & B\ x\ y \\ A & \rightarrow & x\ y\ z \\ & | & q \end{array}$$

   (c)
   $$\begin{array}{rcl} S & \rightarrow & A\ x\ y \\ A & \rightarrow & x\ y\ z \\ & | & \epsilon\ q \end{array}$$

   (d)
   $$\begin{array}{rcl} S & \rightarrow & B\ x\ y \\ A & \rightarrow & x\ y\ z \\ B & \rightarrow & \epsilon \end{array}$$

2. Given the following grammar:

   $$\begin{array}{rcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & F\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

   Which of the following sentences are valid sentences in the language described by the grammar? Give the corresponding parse trees for the valid ones.

   (a) $x\ v\ y$

   (b) $x\ v\ t\ y$

   (c) $x\ v\ p\ v\ y$

   (d) $x\ v\ t\ p\ v\ y$

3. Given the following grammar:

   $$\begin{array}{rcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

   Which of the following sentences are valid sentences in the language described by the grammar? Give the corresponding parse trees for the valid ones.

(a) $x\ v\ y$

(b) $x\ v\ t\ y$

(c) $x\ v\ p\ v\ y$

(d) $x\ v\ t\ p\ v\ y$

4. Which of the following grammars are ambiguous? For the ambiguous one, give a sentence and two parse trees to demonstrate the ambiguity.

(a)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

(b)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ E \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

(c)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & a\ b \\ & | & a\ F \\ F & \rightarrow & b \end{array}$$

(d)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & a\ b \\ & | & a\ E \\ F & \rightarrow & b \end{array}$$

5 What's Next

Once you've understood this guide, there are four others that cover different aspects of grammars that are important for CS 440.

- *Right Linear Grammars* — This guide shows that the context free languages are strictly more expressive than the regular languages. This is proved by showing how any DFA can be converted into a context-free grammar of a special form; and by giving examples of context-free languages that cannot be converted into any DFA.
- *First and Follow Sets* — These sets are important for solving the parsing problem. Both the *LL Parsing* and the *LR Parsing* guides depend on this.
- *LL Parsing* — This guide shows how to make a *recursive descent parser*. This only works if the grammar has certain properties. This guide explains those properties, and methods for restoring those properties to grammars that do not have them.
- *LR Parsing* — LL Parsers have some severe restrictions about the grammars they can accept. The LR Parsing technique circumvents these restrictions, but this technique is much more complicated. Fortunately, there are many good tools that automate this process. This guide shows how to create the parsing tables for an LR parser.

References

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, 1st edition, January 1986.

6 Solutions

1. Only one of these grammars is valid.

   (a) This grammar has a context-sensitive rule.

   (b) This grammar is not valid because there is no rule for the non-terminal B. (If you wish, you could consider B to be a terminal symbol.)

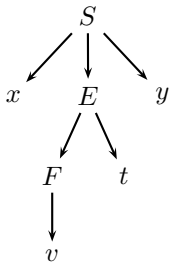
   (c) This grammar is invalid because of the final rule, in which an ϵ occurs together with another symbol. The ϵ should be left out.

   (d) This grammar is fine. True, there are no symbols that would expand the A symbol, but that's not a problem.

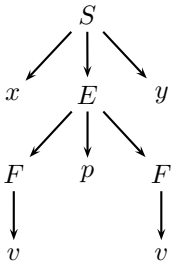
2. Given the grammar:
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & F\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

(a) $x\ v\ y$
This is not a sentence in the language.

(b) $x\ v\ t\ y$



(c) $x\ v\ p\ v\ y$



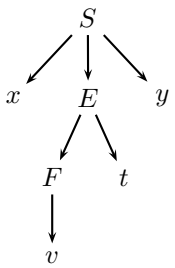
(d) $x\ v\ t\ p\ v\ y$
This is not a sentence in the language.

3. Given the grammar:

$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$

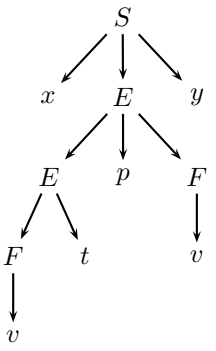
(a) $x\ v\ y$
This is not a sentence in the language.

(b) $x\ v\ t\ y$



(c) $x\ v\ p\ v\ y$
This is not a sentence in the language.

(d) $x\ v\ t\ p\ v\ y$



4. Which of the following grammars are ambiguous? For the ambiguous one, give a sentence and two parse trees to demonstrate the ambiguity.

(a)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ F \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$
Not ambiguous.

(b)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & E\ p\ E \\ & | & F\ t \\ F & \rightarrow & v \end{array}$$
Ambiguous. `x v t p v t p v t y` has two parse trees.

(c)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & a\ b \\ & | & a\ F \\ F & \rightarrow & b \end{array}$$
Ambiguous `x a b y` has two parse trees.

(d)
$$\begin{array}{lcl} S & \rightarrow & x\ E\ y \\ E & \rightarrow & a\ b \\ & | & a\ E \\ F & \rightarrow & b \end{array}$$
Not ambiguous.